Numerical Solution of Partial Differential Equations by using Modified Artificial Neural Network

$Eman\ A. Hussian^{1}$ $Mazin\ H. Suhhiem^{2*}$
1. Dep. of Mathematics, College of Sciences, AL-Mustansiriyah University, Baghdad,Iraq.
2. Dep. of statistics, College of Adm. and Econ., University of sumar, Alrefiey,Iraq.

Abstract
In this paper, we introduce a novel approach based on modified artificial neural network and optimization teqnique to solve partial differential equations. Using modified artificial neural network makes that training points should be selected over an open interval without training the network in the range of first and end points. Therefore, the calculating volume involving computational error is reduced. In fact, the training points depending on the distance selected for training neural network are converted to similar points in the open interval by using a new approach, then the network is trained in these similar areas. In comparison with existing similar neural networks proposed model provides solutions with high accuracy. The proposed method is illustrated by two numerical examples.
Keywords: Partial differential equation, Modified neural network, Feed-forward neural network,BFGS Teqnique, Hyperbolic tangent activation function.

1.Introduction

Differential equations are used as a powerful tool in solving many problems in various fields of human knowledge, such as physics, chemistry, mechanics, economics, etc. one application of the differential equation is turning problems and natural phenomena into differential equations, then by solving the DE the answer is described and the phenomena are calculated. Usually many of these problems do not have analytical solutions or their solution may have certain implications. Many researchers have tried to approximate the solutions of these equations and proposed a lot of algorithms such as: predictor-corrector, Runge-Kutta, finite difference, finite element and other methods. In recent years artificial neural networks for estimation of the ordinary differential equation (ODE) and partial differential equation (PDE) have been used. lee and Kang (1990) used parallel processor computers to solve a first order differential equation with Hopfield neural network models. Meade and Fernandez (1994) solved linear and non-linear ODE by using feed-forward neural networks architecture and B-splines of degree one. Lagaris, Likas & Fotiadis (1998) used artificial neural network for solving ODEs and PDEs with the initial / boundary value problems. In comparison with jammes and Liu (Liu & Jammes 1999), malek and shekari presented numerical method based on neural network and optimization techniques which the higher-order DE answers approximates by finding a package form analytical of specific functions (Malek & Shekari 2006) which is a combination of two terms:
The first term is related to the initial / boundary condition and the second term contains parameters related to the neural network, the used neural network is a two-layer network with one hidden layer and the activation function used in the hidden layer is a sigmoid function or hyperbolic tangent function. During the past few years, the numerical solutions of ODEs and PDEs by using artificial neural network have been studied by several authors (Ahmed & Bilal 2014, Biglari,Assareh , Poultangari & Nedaei 2013, Baymani,Kerayechian & Effaati 2010, Ghalambaz,Noghrehabadi,Behrang,Assareh,Ghanbarzad & Hedayat 2011 , Ibraheem & Khalaf 2011,Jalab , Ibrahim ,Murad,Mulhum & Hadid 2012, Tawfiq & Al-Abrahemee 2014, Putcha & Deepika 2013, Pattanaik & Mishra 2008, Parand,Roozbahani & Babolghani 2013, Wu & Hsu 2012 Yazdi,Pakdaman & Modaghegh 2011).
In this paper we view the problem from a different angle. We present a modified method for solving partial differential equations (PDE's) that relies on the function approximation capabilities of feed forward neural networks and results in the construction of a solution written in a differentiable, closed analytic form. This form employs a feed forward neural network as the basic approximation element, whose parameters (weights and biases) are adjusted to minimize an appropriate error function. To train the ANN which we design, we employ optimization techniques, which in turn require the computation of the gradient of the error with respect to the network parameters. In the proposed approach the model function is expressed as the sum of the two terms: the first term satisfies the initial / boundary conditions and contains no adjustable parameters. The second term can be found by using feed forward neural network(FFNN), which is trained so as to satisfy the differential equation. Since it is known that a multilayer FFNN with one hidden layer can approximate any function to arbitrary accuracy, thus our ANN contains one hidden layer.
This modified method is called modified artificial neural network(MANN) for solving ordinary and partial differential equations. This new method based on replaced every x in the training set (where $x \in [a, b]$) by the polynomial $Q(x) = \epsilon (x + 1)$ such that $Q(x) \in (a, b)$ by choosing a suitable $\epsilon \in (0, 1)$. In this paper, we will illustrate this modified method by solving *two* numerical examples and we will introduce a comparison between

our results and a results which are calculated by other numerical methods such as finite element method.

2. Artificial Neural Network

An Artificial neural network (ANN) is a simplified mathematical model of the human brain, it can be implemented by both electric elements and computer software. It is a parallel distributed processor with large numbers of connections, it is an information processing system that has certain performance characters in common with biological neural networks. ANN have been developed as generalizations of mathematical models of human cognition or neural biology, based on the assumptions:

1. lnformation processing occurs at many simple elements called neurons that is fundamental the operation of ANN's.
2. Signals are passed between neurons over connection links.
3. Each connection link has an associated weight which, in a typical neural net, multiplies the signal transmitted.
4. Each neuron applies an activation function (usually nonlinear) to its net input (sum of weighted input signals) to determine its output signal.

There are two main connection formulas (types):feedback(recurrent) and feed-forward connections.Feedback is one type of connection where the output of one layer routes back to the input of a previous layer , or to the same layer.Feed-forward neural network(FFNN) does not have a connection back from the output to the input neurons(Fig.1).There are many different training algorithms, but the most often used training algorithm is the back propagation(BP)rule.ANN is trained to map a set of input data by iterative adjusment of the weights.Information from inputs is feedforward through the network to optimize the wieghts between neurons. Optimization of the wieghts is made by backward propagation of the error during training phase.The ANN reads the input and output values in the training data set and changes the value of the wieghted links to reduce the difference between the predicted and target(observed)values.The error in prediction is minimized across many training cycles(iteration or epoch) until network reaches specified level of accuracy.A complete round of forward backward passes and wieght adjusments using all input output pairs in the data set is called an epoch or iteration. In order to perform a supervised training we need a way of evaluating the ANN output error between the actual and the expected outputs .A popular measure is the mean squared error (MSE) or root mean squared error(RMSE) (Tawfiq 2004, Tawfiq & Al-Abrahemee 2014)

3. Description of The Method

In this section we will illustrate how our approach can be used to find the approximate solution of the general form a second order differential equation(Lagaris,Likas & Fotiadis 1998):

$$G\big(\vec{x}\,,\Psi(\vec{x})\,,\nabla\,\Psi(\vec{x})\,,\nabla^2\,\Psi(\vec{x})\big) = 0\,,\ \ \vec{x}\in D \qquad (1)$$

Where a subject to certain boundary conditions (BC's) or initial conditions (IC's) (for instance Dirichlet and / or Neumann conditions) and $\vec{x} = (x_1\,, x_2\,, \dots\dots, x_n) \in R^n$, $D \subset R^n$ denotes the domain and $\Psi(\vec{x})$ is the solution to be computed.

To obtain a solution to the above differential equation, the collocation method is adopted which assumes a discretization of the domain D and its boundary S into a set points $\widehat{D}$ and $\widehat{S}$, respectively. The problem is then transformed into the following system of equations:

$$G\big(\vec{x}_i\,,\Psi(\vec{x}_i)\,,\nabla\,\Psi(\vec{x}_i)\,,\nabla^2\,\Psi(\vec{x}_i)\big) = 0\,,\ \forall\,\vec{x}_i \in \widehat{D} \qquad (2)$$

Subject to the constraints imposed by the BC's or IC's.

If $\Psi_t\ (\vec{x}\,,\vec{p})$ denotes a trial solution with adjustable parameters $\vec{p}$, the problem is transformed to a discretize form

$$\underset{\vec{p}}{\text{Min}}\ \sum_{\vec{x}_i\,\in\,\widehat{D}} \Big(G\big(\vec{x}_i\Psi_t(\,\vec{x}_i,\vec{p}\,),\nabla\Psi_t(\,\vec{x}_i,\vec{p}\,),\nabla^2\Psi_t(\,\vec{x}_i,\vec{p}\,)\big)\Big)^2 \qquad (3)$$

Subject to the constraints imposed by the BC's or IC's.

In the our proposed approach, the trial solution Ψ_t employs a feed forward neural network and the parameters $\vec{p}$ correspond to the weights and biases of the neural architecture. We choose a form for the trial function $\Psi_t(\vec{x})$ such that it satisfies the BC's or IC's. This is achieved by writing it as a sum of two terms.

$$\Psi_t(\vec{x}) = A(\vec{x}) + F\big(\vec{x}\,, N(\vec{x}\,,\vec{p})\big) \qquad (4)$$

Where $N(\vec{x}\,,\vec{p})$ is a single-output feed forward neural network with parameters $\vec{p}$ and n input units fed with the input vector $\vec{x}$.

The term $A(\vec{x})$ contains no adjustable parameters and satisfies the boundary conditions. The second term F is constructed so as not to contribute to the BC's or IC's, since $\Psi_t(\vec{x})$ satisfy them. This term can be formed by using a ANN whose weights and biases are to be adjusted in order to deal with the minimization problem (Lagaris,Likas & Fotiadis 1998).

Our numerical result shows that our approach, which based on the above formulation is very effective and can be done in reasonable computing time to compute an accurate solutions.

4. Computation of The Gradient

An efficient minimization of (3) can be considered as a procedure of
training the ANN, where the error corresponding to each input vector $\vec{x}_i$ is the value $E(\vec{x}_i)$ which has to forced

near zero. Computation of this error value involves not only the ANN output (as is the case in conventional training) but also the derivatives of the output with respect to any of its inputs. Therefore, in computing the gradient of the error with respect to the network weights (Tawfiq 2004).
Consider a multilayer FFNN with n input units, one hidden layer with H sigmoid units and a linear output unit. The extension to the case of more than one hidden layers can be obtained accordingly.
For a given input vector $\vec{x} = (x_1, x_2, \dots, x_n)$ the output of the ANN is:
$N = \sum_{i=1}^{H} v_i \, s(z_i)$, where $z_i = \sum_{j=1}^{n} w_{ij} \, x_j + b_i$
w_{ij} denotes the weight connecting the input unit j to the hidden unit i,
v_i denotes the weight connecting the hidden unit i to the out put unit,
b_i denotes the bias of hidden unit i, and
$s(z)$ is the hyperbolic tangent activation function.
The gradient of ANN, N with respect to the parameters of the ANN can be easily obtained as:

$$\frac{\partial N}{\partial v_i} = s(z_i) \qquad (5)$$

$$\frac{\partial N}{\partial b_i} = v_i \, s'(z_i) \qquad (6)$$

$$\frac{\partial N}{\partial w_{ij}} = v_i \, s'(z_i) x_j \qquad (7)$$

Once the derivative of the error with respect to the network parameters has been defined, then it is a straight forward to employ any minimization technique and we will use BFGS quasi-Newton method .

5. Illustration of The Method

We will consider a two - dimensional problems. However, it is straightforward to extend the method to more dimensions.
For example, consider the Poisson equation (Lagaris,Likas & Fotiadis 1998):

$$\frac{\partial^2 \Psi(x,y)}{\partial x^2} + \frac{\partial^2 \Psi(x,y)}{\partial y^2} = f(x,y) \qquad (8)$$

$x \in [0,1]$, $y \in [0,1]$ with Dirichlet BC:
$\Psi(0,y) = f_0(y)$, $\Psi(1,y) = f_1(y)$, $\Psi(x,0) = g_0(x)$ and $\Psi(x,1) = g_1(x)$, where f_0 , f_1 , g_0 and g_1 are continuous function.
The trial solution is written as

$$\Psi_t(x,y) = A(x,y) + x(1-x)\, y(1-y)\, N(x,y,\vec{p}) \qquad (9)$$

Where $A(x,y)$ is chosen so as to satisfy the BC, namely:

$$A(x,y) = (1-x)\, f_0(y) + x\, f_1(y) + (1-y)\left\{g_0(x) - \left[(1-x)\, g_0(0) + x g_0(1)\right]\right\} + y\left\{g_1(x) - \left[(1-x)\, g_1(0) + x g_1(1)\right]\right\} \qquad (10)$$

For mixed boundary conditions of the form:
$\Psi(0,y) = f_0(y)$, $\Psi(1,y) = f_1(y)$, $\Psi(x,0) = g_0(x)$ and $(\partial\Psi(x,1)/\partial y) = g_1(x)$
(i.e., Dirichlet on part of the boundary and Neumann elsewhere), the trial solution can be written as

$$\Psi_t(x,y) = B(x,y) + x(1-x)y[N(x,y,\vec{p}) - N(x,1,\vec{p}) - \partial N(x,1,\vec{p})/\partial y] \qquad (11)$$

And $B(x,y)$ is again chosen so as to satisfy the BC's:

$$B(x,y) = (1-x)f_0(y) + x\, f_1(y) + g_0(x) - \left[(1-x)\, g_0(0) + x g_0(1)\right] + y\left\{g_1(x) - \left[(1-x)\, g_1(0) + x g_1(1)\right]\right\} \qquad (12)$$

Note that the second term of the trial solution does not affect the boundary conditions since it vanishes at the part of the boundary where Dirichlet BC's are imposed and its gradient component normal to the boundary vanishes at the part of the boundary where Neumann BC's are imposed. In all the above PDE problems the error that should be minimized is given by:

$$E[\vec{p}] = \sum_{i=1}^{n}\left\{\frac{\partial^2 \Psi_t(x_i,y_i)}{\partial x^2} + \frac{\partial^2 \Psi_t(x_i,y_i)}{\partial y^2} - f(x_i,y_i)\right\}^2 \qquad (13)$$

Where (x_i, y_i) are points in $[0,1] \times [0,1]$ (Tawfiq 2004).

6. Proposed Method

In this section we will introduce a novel method to modify the artificial neural network ANN . This new method based on replaced every x in the input vector (training set) $\vec{x} = (x_1, x_2, \dots\dots, x_n)$, $x_j \in [a,b]$ by a polynomial of degree one. Ezadi and parandin (2013) used the function:

$$Q(x) = \epsilon\,(x+1)\,,\ \epsilon \in (0,1) \qquad (14)$$

Then the input vector will be: $(Q(x_1), Q(x_2), \dots\dots Q(x_n))$, $Q(x_j) \in (a,b)$ In this paper, we named this proposed method modified artificial neural network (MANN). Using modified artificial neural network makes that training points should be selected over the open interval (a,b) without training the neural network in the range of first and end points. therefore, the calculating volume involving computational error is reduced. In fact, the training points depending on the distance [a , b] selected for training neural network are converted to similar points in the open interval (a , b) by using a new approach, then the network is trained in these similar areas. From above, we have: for a given input vector $(x_1, x_2, \ldots\ldots, x_n)$, $x_j \in [a, b]$, the output of the modified artificial neural network is:

$N = \sum_{i=1}^{H} v_i\ s(z_i)$, where $z_i = \sum_{j=1}^{n} W_{ij}\ Q(x_j) + b_i$ (15)

and $Q(x_j) = \epsilon\ (x_j + 1)$, $\epsilon \in (0,1)$ and $x_j \in [a , b]$, then $Q(x_j) \in (a , b)$

Note that, equations (5 _ 7) will be:

$\frac{\partial N}{\partial v_i} = s\big(W_{ij}\ Q(x_j) + b_i\big) = s\big(\epsilon\ (x_j + 1)\ W_{ij} + b_i\big)$ (16)

$\frac{\partial N}{\partial b_i} = v_i\ s'\big(W_{ij}\ Q(x_j) + b_i\big) = v_i\ s'\big(\epsilon\ (x_j + 1)\ W_{ij} + b_i\big)$ (17)

$\frac{\partial N}{\partial w_{ij}} = v_i\ Q(x_j)\ s'\big(W_{ij}\ Q(x_j) + b_i\big) = v_i\ Q(x_j)\ s'\big(\epsilon\ (x_j + 1)\ W_{ij} + b_i\big)$ (18) where s' is the first derivative of the hyperbolic tangent function.

7. Solution of Partial Differential Equations

We will consider boundary value problems partial differential equation with Dirichlet conditions or Neumann conditions. All the subsequent problems were defined on the domain [0,1] × [0,1] and in order to perform training we consider a mesh points obtained by considering ten equidistant points of the domain [0,1] of each variable. In analogy with the previous cases the neural network archilecture was considered to be FFNN with two inputs (accepting the coordinates x and y of each point), 10 hidden units and one linear output unit (Fig.1).

For every entries x and y , the input neurons makes no changes in its inputs, so the input to the hidden neurons is:

$Net_j = x\, w_{j1} + y\, w_{j2} + B_j$, $j = 1,2,\ldots\ldots,m$ (19)

Where w_{j1} and w_{j2} are a weights from the input layer to the jth unit in the hidden layer, B_j is an jth bias for the jth unit in the hidden layer. The output in the hidden neurons is :

$Z_j = s(net_j)$, $j = 1,2,\ldots\ldots,m.$ (20)

The output neuron make no changes in its input, so the input to the output neuron is equal to output: $N = \sum_{j=1}^{m} V_j\ Z_j$. …(21)

• For MANN, eq.(19) will be:

$net_j = Q(x)\ w_{j1} + Q(y)\ w_{j2} + B_j = \epsilon(x+1)\ w_{j1} + \epsilon(y+1)\ w_{j2} + B_j$ (22) $Z_j = s(\epsilon(x+1)\ w_{j1} + \epsilon(y+1)\ w_{j2} + B_j)$ (23)

$N = \sum_{j=1}^{m} V_j\ s(\epsilon(x+1)\ w_{j1} + \epsilon(y+1)\ w_{j2} + B_j)$ (24)

Where $j = 1,2,\ldots\ldots,m$, $\epsilon \in (1,0)$ and $Q(x)$, $Q(y) \in (a, b)$

8. Numerical Examples

In the section we report some numerical results and the solution of a number of model problems. In all cases we used a three-layer FFNN having two input units, one hidden layer with 10 hidden units (neurons) and one output unit, and hyperbolic tangent activation function, that is: $s(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. For each test problem, the analytical solution $u_a(\vec{x})$ was known in advance, therefore we test the accuracy of the obtained solutions by computing the deviation: $\Delta u(\vec{x}) = \left|u_t(\vec{x}) - u_a(\vec{x})\right|$ (25)

We will use BFGS quasi-Newton method to minimize the error function. Also, we will introduce a comparison between the usual artificial neural network(UANN) and the modified artificial neural network(MANN) by using a numerical results in other references .

Example (8.1): Consider the boundary value problem:

$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$, $x, y \in [0,1]$

With the Dirichlet boundary conditions:

$u(0, y) = 0$, $u(1, y) = 0$, $u(x, 0) = 0$ and $u(x, 1) = \sin \pi x$.

the analytical solution is: $u_a(x, y) = \frac{\sin\pi x \sinh\pi y}{\sinh\pi}$.

By using (9), the trial solution has the form:

$u_t(x, y) = y \sin \pi x + xy\ (x - 1)\ (y - 1)\ N\ (x , y, p)$.

To find the error function E that must be minimized we calculate:

$\frac{\partial^2 u_t(x,y)}{\partial x^2} = -\pi^2 y \sin \pi x + (y^2 - y)\left[\left(x^2 - x\right)\frac{\partial^2 N}{\partial x^2} + \left(4x - 2\right)\frac{\partial N}{\partial x} + 2N\right]$ (26)

$\frac{\partial^2 u_t(x,y)}{\partial y^2} = \left(x^2 - x\right)\left[\left(y^2 - y\right)\frac{\partial^2 N}{\partial y^2} + \left(4y - 2\right)\frac{\partial N}{\partial y} + 2N\right]$ (27)

By using eq. (13), we have:

$$E(p) = \sum_{i=1}^{11} \begin{bmatrix} -\pi^2 y_i \sin \pi x_i + (y_i^2 - y_i)\left((x_i^2 - x_i)\frac{\partial^2 N(x_i, y_i, p)}{\partial x^2}\right) + (4x_i - 2) \\ \frac{\partial N(x_i, y_i, p)}{\partial x} + 2N(x_i, y_i, p) + (x_i{}^2 - x_i) \\ \left((y_i^2 - y_i)\frac{\partial^2 N(x_i, y_i, p)}{\partial y^2} + (4y_i - 2)\frac{\partial N(x_i, y_i, p)}{\partial x} + 2N(x_i, y_i, p)\right) \end{bmatrix}^2 \quad (28)$$

For eq. (21), we have:

$N(x, y, p) = \sum_{j=1}^{10} v_j\, s\,(x\, w_{j1} + y\, w_{j2} + B_j)$ (29)

$\frac{\partial N(x,y,p)}{\partial x} = \sum_{j=1}^{10} v_j\, w_{j1}\, s'\,(x\, w_{j1} + y\, w_{j2} + B_j)$ (30)

$\frac{\partial^2 N(x,y,p)}{\partial x^2} = \sum_{j=1}^{10} v_j\, w_{j1}^2\, s''\,(x\, w_{j1} + y\, w_{j2} + B_j)$ (31)

$\frac{\partial N(x,y,p)}{\partial y} = \sum_{j=1}^{10} v_j\, w_{j1}\, s'\,(x\, w_{j1} + y\, w_{j2} + B_j)$ (32)

$\frac{\partial^2 N(x,y,p)}{\partial y^2} = \sum_{j=1}^{10} v_j\, w_{j1}^2\, s''\,(x\, w_{j1} + y\, w_{j2} + B_j)$ (33)

By substitute (29 – 33) in (28) , we have:

$$E = \sum_{i=1}^{11} \begin{bmatrix} -\pi^2 y_i \sin \pi x_i + (y_i^2 - y_i)\,((x_i^2 - x_i)\sum_{j=1}^{10} v_j\, w_{j1}^2 \\ s''(x_i\, w_{j1} + y\, w_{j2} + B_j) + (4x_i - 2)\sum_{j=1}^{10} v_j\, w_{j1} \\ s'(x_i\, w_{j1} + y\, w_{j2} + B_j) + 2\sum_{j=1}^{10} v_j\, s\,(x_i\, w_{j1} + y_i\, w_{j2} + B_j)) \\ +(x_i^2 - x_i)((y_i^2 - y_i)\sum_{j=1}^{10} v_j\, w_{j2}^2\, s''\,(x_i\, w_{j1} + y_i\, w_{j2} + B_j) \\ +(4y_i - 2)\sum_{j=1}^{10} v_j\, w_{j2}\, s'\,(x_i\, w_{j1} + y_i\, w_{j2} + B_j) + 2 \\ \sum_{j=1}^{10} v_j\, s\,(x_i\, w_{j1} + y_i\, w_{j2} + B_j)\,) \end{bmatrix}^2 \quad (34)$$

Then one can use (34) to adjust the weights and biases with respect to usual artificial neural network(UANN).

• For MANN : in the same way, we have:

$$E = \sum_{i=1}^{11} \begin{bmatrix} -\pi^2 y_i \sin \pi x_i + (y_i^2 - y_i)\,((x_i^2 - x_i)\sum_{j=1}^{10} v_j\, w_{j1}^2\ s'' \\ (Q(x_i)\, w_{j1} + Q(y_i)\, w_{j2} + B_j) + (4x_i - 2)\sum_{j=1}^{10} v_j\, w_{j1} s' \\ (Q(x_i)\, w_{j1} + Q(y_i)\, w_{j2} + B_j) + 2\sum_{j=1}^{10} v_j s \\ (Q(x_i)\, w_{j1} + Q(y_i)\, w_{j2} + B_j)) + (x_i^2 - x_i)((y_i^2 - y_i) \\ \sum_{j=1}^{10} v_j\, w_{j2}^2\, s''\,(Q(x_i)\, w_{j1} + Q(y_i)\, w_{j2} + B_j) + (4y_i - 2) \\ \sum_{j=1}^{10} v_j\, w_{j2}\, s'\,(Q(x_i)\, w_{j1} + Q(y_i)\, w_{j2} + B_j) + 2 \\ \sum_{j=1}^{10} v_j\, s\,(Q(x_i)\, w_{j1} + Q(y_i)\, w_{j2} + B_j)\,) \end{bmatrix}^2 \quad (35)$$

We use (35) to adjust the weight and biases w. r. t. MANN.

Since $x, y \in [0,1]$, then $Q(x), Q(y) \in (0,1)$ and since $Q(x) = \epsilon(x+1)$, $Q(y) = \epsilon(y+1)$, then we must choose $\epsilon < 0.5$. for $\epsilon = 0.4$, the training set will be:

x :	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y :	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Q(x):	0.4	0.44	0.48	0.52	0.56	0.6	0.64	0.68	0.72	0.76	0.8
Q(y):	0.4	0.44	0.48	0.52	0.56	0.6	0.64	0.68	0.72	0.76	0.8

Also , we solved this example by using usual artificial neural network, these result can be found in table (1). Analytical and trial solutions for this problem can be found in table (1) and figures (2) and (3) .

Example (8. 2): Consider the non-linear PDE:

$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + u\,\frac{\partial u}{\partial y} = \sin \pi x\,(2 - \pi^2 y^2 + 2y^3 \sin \pi x),$

Where $x, y \in [0,1]$ and with mixed boundary conditions:

$u(0,y) = 0$, $u(1,y) = 0$, $u(x, 0) = 0$, $\frac{\partial u}{\partial y}(x,1) = 2 \sin \pi x.$

The analytical solution for this problem is: $u_a(x, y) = y^2 \sin \pi x$.

By using (11 and 12) , the trial solution has the form:

$u_t(x, y) = 2y \sin \pi x + xy\,(1 - x)\left[N(x, y, p) - N(x, 1, p) - \frac{\partial N(x,1,p)}{\partial y}\right]$ (36)

From (36), we can find:

$$\frac{\partial^2 u_t(x,y)}{\partial x^2} = -2\pi^2 y \sin \pi x + y(x - x^2)\left(\frac{\partial^2 N(x,y,p)}{\partial x^2} - \frac{\partial^2 N(x,1,p)}{\partial x^2} - \frac{\partial^3 N(x,1,p)}{\partial y\, \partial x\, \partial x}\right) + 2y(1\text{-}2x)\left(\frac{\partial N(x,y,p)}{\partial x} - \frac{\partial N(x,1,p)}{\partial x} - \frac{\partial^2 N(x,1,p)}{\partial y\, \partial x}\right) - 2y\left(N(x,y,p) - N(x,1,p) - \frac{\partial N(x,1,p)}{\partial y}\right) \quad (37)$$

$$\frac{\partial u_t(x,y)}{\partial y} = 2 \sin \pi x + (x - x^2)\left(y \frac{\partial N(x,y,p)}{\partial y} + N(x,y,p) - N(x,1,p) - \frac{\partial N(x,1,p)}{\partial y}\right) \quad (38)$$

$$\frac{\partial^2 u_t(x,y)}{\partial y^2} = (x - x^2)\left(y \frac{\partial^2 N(x,y,p)}{\partial y^2} + 2\frac{\partial N(x,y,p)}{\partial y}\right) \quad (39)$$

Form eq. (13) , we have:

E=

$$\sum_{i=1}^{11}\left[\begin{array}{c} -2\pi^2 y_i \sin \pi x_i + y_i(x_i - x_i^2) \\ \left(\frac{\partial^2 N(x_i,y_i,p)}{\partial x^2} - \frac{\partial^2 N(x_i,1,p)}{\partial x^2} - \frac{\partial^3 N(x_i,1,p)}{\partial y\, \partial x\, \partial x}\right) + 2y_i\,(1 - 2x_i) \\ \left(\frac{\partial N(x_i,y_i,p)}{\partial x} - \frac{\partial N(x_i,1,p)}{\partial x} - \frac{\partial^2 N(x_i,1,p)}{\partial y\, \partial x}\right) - 2y_i \\ \left(N(x_i,y_i,p) - N(x_i,1,p) - \frac{\partial N(x_i,1,p)}{\partial y}\right) + (x_i - x_i^2) \\ \left(y_i \frac{\partial^2 N(x_i,y_i,p)}{\partial y^2} + 2\frac{\partial N(x_i,y_i,p)}{\partial y}\right) + \\ \left(2y_i \sin \pi x_i + y_i(x_i - x_i^2)\left(N(x_i,y_i,p) - N(x_i,1,p) - \frac{\partial N(x_i,1,p)}{\partial y}\right)\right) * \\ \left(2 \sin \pi x_i + (x_i - x_i^2)\left(y_i \frac{\partial N(x_i,y_i,p)}{\partial y} + N(x_i,y_i,p) - N(x_i,1,p) - \frac{\partial N(x_i,1,p)}{\partial y}\right)\right) \\ - \sin \pi x_i\,(2 - \pi^2 y_i^2 + 2y_i^3 \sin \pi x_i) \end{array}\right]^2 \quad (40)$$

Where :

$$N(x,1,p) = \sum_{j=1}^{10} v_j\, s\,(x\, w_{j1} + w_{j2} + B_j) \quad (41)$$

$$\frac{\partial N(x,1,p)}{\partial x} = \sum_{j=1}^{10} v_j\, w_{j1}\, s'\,(x\, w_{j1} + w_{j2} + B_j) \quad (42)$$

$$\frac{\partial^2 N(x,1,p)}{\partial x^2} = \sum_{j=1}^{10} v_j\, w_{j1}^2\, s''\,(x\, w_{j1} + w_{j2} + B_j) \quad (43)$$

$$\frac{\partial N(x,1,p)}{\partial y} = \sum_{j=1}^{10} v_j\, w_{j2}\, s'\,(x\, w_{j1} + w_{j2} + B_j) \quad (44)$$

$$\frac{\partial^2 N(x,1,p)}{\partial y\, \partial x} = \sum_{j=1}^{10} v_j w_{j1}\, w_{j2}\, s''\,(x\, w_{j1} + w_{j2} + B_j) \quad (45)$$

$$\frac{\partial^3 N(x,1,p)}{\partial y\, \partial x\, \partial x} = \sum_{j=1}^{10} v_j\, w_{j1}^2 w_{j2}\, s'''\,(x\, w_{j1} + w_{j2} + B_j) \quad (46)$$

• For MANN : in the same way, we have:

$$E = \sum_{i=1}^{11}\left[\begin{array}{c} -2\pi^2 y_i \sin \pi x_i + y_i(x_i - x_i^2) \\ \left(\frac{\partial^2 N(Q(x_i),Q(y_i),p)}{\partial x^2} - \frac{\partial^2 N(Q(x_i),1,p)}{\partial x^2} - \frac{\partial^3 N(Q(x_i),1,p)}{\partial y\, \partial x\, \partial x}\right) + 2y_i(1 - 2x_i) \\ \left(\frac{\partial N(Q(x_i),Q(y_i),p)}{\partial x} - \frac{\partial N(Q(x_i),1,p)}{\partial x} - \frac{\partial^2 N(Q(x_i),1,p)}{\partial y\, \partial x}\right) - 2y_i \\ \left(N(Q(x_i),Q(y_i),p) - N(Q(x_i),1,p) - \frac{\partial N(Q(x_i),1,p)}{\partial y}\right) + (x_i - x_i^2) \\ \left(y_i \frac{\partial^2 N(Q(x_i),Q(y_i),p)}{\partial y^2} + 2\frac{\partial N(Q(x_i),Q(y_i),p)}{\partial y}\right) + \\ \left(2y_i \sin \pi x_i + y_i(x_i - x_i^2)\begin{pmatrix} N(Q(x_i),Q(y_i),p) - N(Q(x_i),1,p) - \\ \frac{\partial N(Q(x_i),1,p)}{\partial y} \end{pmatrix}\right) * \\ \left(2 \sin \pi x_i + (x_i - x_i^2)\begin{pmatrix} y_i \frac{\partial N(Q(x_i),Q(y_i),p)}{\partial y} + N(Q(x_i),Q(y_i),p) - \\ N(Q(x_i),1,p) - \frac{\partial N(Q(x_i),1,p)}{\partial y} \end{pmatrix}\right) - \\ \sin \pi x_i\,(2 - \pi^2 y_i^2 + 2y_i^3 \sin \pi x_i) \end{array}\right]^2 \quad (47)$$

where

$$N(Q(x),1,p) = \sum_{j=1}^{10} v_j\, s\left(Q(x)\, w_{j1} + w_{j2} + B_j\right) \qquad (48)$$
$$\frac{\partial\, N(Q(x),1,p)}{\partial x} = \sum_{j=1}^{10} v_j\, w_{j1}\, s'\left(Q(x)\, w_{j1} + w_{j2} + B_j\right) \qquad (49)$$
$$\frac{\partial^2\, N(Q(x),1,p)}{\partial x^2} = \sum_{j=1}^{10} v_j\, w_{j1}^2\, s''\left(Q(x)\, w_{j1} + w_{j2} + B_j\right) \qquad (50)$$
$$\frac{\partial\, N(Q(x),1,p)}{\partial y} = \sum_{j=1}^{10} v_j\, w_{j2}\, s'\left(Q(x)\, w_{j1} + w_{j2} + B_j\right) \qquad (51)$$
$$\frac{\partial^2\, N(Q(x),1,p)}{\partial y\, \partial x} = \sum_{j=1}^{10} v_j w_{j1}\, w_{j2}\, s''\left(Q(x)\, w_{j1} + w_{j2} + B_j\right) \qquad (52)$$
$$\frac{\partial^3\, N(Q(x),1,p)}{\partial y\, \partial x\, \partial x} = \sum_{j=1}^{10} v_j\, w_{j1}^2 w_{j2}\, s'''\left(Q(x)\, w_{j1} + w_{j2} + B_j\right) \qquad (53)$$

Then we can use (47) to update the weights and biases w.r.t. MANN.
Since $x, y \in [0,1]$, then Q(x) , Q(y) $\in (0,1)$. Therefore, we must choose $\epsilon < 0.5$. for $\epsilon = 0.2$, the training set will be:

x:	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y:	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Q(x):	0.2	0.22	0.24	0.26	0.28	0.3	0.32	0.34	0.36	0.38	0.4
Q(y):	0.2	0.22	0.24	0.26	0.28	0.3	0.32	0.34	0.36	0.38	0.4

Lagaris,Likas & Fotiadis(1998) solved this problem by using UANN, the maximum deviation in these result ≈ 0.000015, we compared this absolute error with the absolute error which we found by using MANN. Analytical and trial solutions for this problem can be found in table (2) and figures (4) and (5) .

9. Results and Discussion

From the numerical examples in this work , it is clear that the modified artificial neural network gives best results and better accuracy in comparison with usual artificial neural network . We can conclude that the method we propused can handle effectively all types of partial differential equations and provide accurate approximate solution throughout the whole domain and not only at the training set. Therefore, one can use the interpolation techniques (such as curve fitting method) to find the approximate solution at points between the training points or at points outside the training set.

Table(3)reports the maximum and minimum absolute error corresponding to the MANN and to the UANN .

Note that for example (8.2), Lagaris, Likas & Fotiadis (1998) they had compared the results obtained with the result that found by using the finite element method which has been widely acknowledged as one of the most effective approaches to find the approximate solutions of differential equations. The maximum absolute error by using FEM is 0.0000006, this result is better than that obtained by using the MANN which is 0.0000024, but we must Note that the accuracy of the MANN method can be controlled by increasing the number of hidden units or expansion the training set.

10. Conclusion

In this paper, we presented a hybrid approach based on modified artificial neural networks for solving partial differential equations. We demonstrate, for the first time, the ability of modified artificial neural networks to approximate the solutions of PDEs . The main reason for using modified artificial neural networks was their applicability in function approximation. Further research is in progress to apply and extend this method to solve three-dimensional partial differential equations and integral equations .

References

Ahmed,I. & Bilal ,M.(2014)," Numerical Solution of Blasius Equation Through Neural Networks Algorithm " , American Journal of Computational Mathematics,4,223-232.

Biglari ,M. , Assareh,E. , Poultangari ,I. & Nedaei ,M. (2013) ," Solving Blasius Differential Equation by Using Hybrid Neural Network and Gravitational Search Algorithm (HNNGSA)",Global Journal of Science , Engineering and Technology,11,29-36.

Baymani, M., Kerayechian, A. & Effaati ,A.(2010),"Artificial Neural Networks Approach for Solving Stokes Prpblem ", Applied Mathematics,1,288-292.

Ezadi,S. & Parandin,N.(2013),"An Application of Neural Networks to Solve Ordinary Differential Equations",International Journal of Mathematical Modelling & Computations,3(3),245-252.

Ghalambaz,M.,Noghrehabadi,A.R.,Behrang,M.,Assareh,E.,Ghanbarzad,A. & Hedayat , N. (2011) , " A Hybrid Neural Network and Gravitational Search Algorithm (HNNGSA) Method to Solve Well Known Wessinger's Equation", World Academy of science,Engineering and Technology,5,1-21.

Ibraheem,K.I. & Khalaf,B.M.(2011),"Shooting Neural Networks Algorithm for Solving Boundary Value Problems in ODEs",Applications and Applied Mathematics:An International Journal(AAM),11,1927-1941.

Jalab , H. A. Ibrahim , R.W., Murad , S .A . , Mulhum ,A.I. & Hadid, S. B. (2012)," Numerical Solution of Lane-Emden Equation Using Neural Network" , International Conference on Fundamental and

Applied Sciences,1482,414-418.

Liu,B. & Jammes ,B.(1999)," Solving Ordinary Differential Equations by Neural Networks", in:Proceeding of 13th European Simulation Multi- Conference Modelling and Simulation: A Tool for the Next Millennium , Warsaw,Poland,June 14.

Lee,H. & Kang,I.S.(1990),"Neural Algorithms for Solving Differential Equations", Journal of Computational Physics,91,110-131.

Lagaris,I.E.,Likas,A. & Fotiadis ,D.I.(1998)," Artificial Neural Networks for Solving Ordinary and Partial Differential Equations",IEEE Transactions on Neural Networks,9(5),987-1000.

Meade,A.J. & Fernandez,A.A.(1994),"Solution of Linear Ordinary Differential Equations By Feed-Forward Neural Networks",Mathematical and Computer Modelling,19(12),1-25.

Meade,A.J. & Fernandez,A.A.(1994),"Solution of Non-Linear Ordinary Differential Equations By Feed-Forward Neural Networks",Mathematical and Computer Modelling,20(9),19-44.

Malek,A. & Shekari,R.(2006),"Numerical Solution for High Order Differential Equations,Using a Hybrid Neural Network Optimization Method",Applied Mathematics and Computation,183,260-271.

Putcha,V.S. & Deepika,G.V.(2013),"Continuous Lyapunov Dynamical Systems-Artificial Neural Network Approach",International Journal of Differential Equations and Applications,12(4),139-149.

Pattanaik,S. & Mishra,R.K.(2008),"Application of ANN for Solution of PDE In RF Engineering",International Journal on Information Sciences and Computing,2(1),74-79.

Parand,K.,Roozbahani,Z. & Babolghani , F.P.(2013), "Solving Non-Linear Lane-Emden Type Equations With Unsupervised Combined Artificial Neural Networks", International Journal Industrial Mathematics,5(4),355-366.

Tawfiq,L.N.M.(2004),"On Design and Training of Artificial Neural Networks for Solving Differential Equations".PH.D Thesis,52-65.

Tawfiq,L.N.M. & Al-Abrahemee,K.M.M.(2014),"Design Neural Network to Solve Singular Perturbation Problems",Applied & Computational Mathematics,3.

Wu,S.L. & Hsu,C.H.(2012),"Entire Solutions of Non-Linear Cellular Neural Networks With Distributed Time Delays",Nonlinearity, 25,2785-2801.

Yazdi,H.S.,Pakdaman,M. & Modaghegh,H.(2011), " Unsupervised Kernal Least Mean Square Algorithm for Solving Ordinary Differential Equations",Neurocomputing ,74,2062-2071.

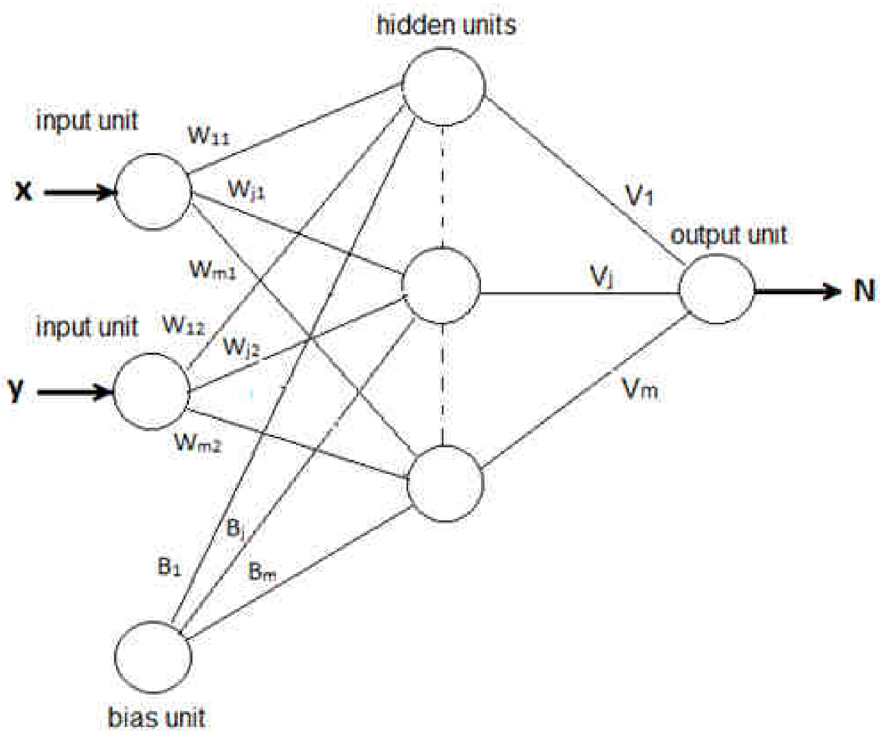

Figure 1. (2 x m x 1) Feed-forword neural network.

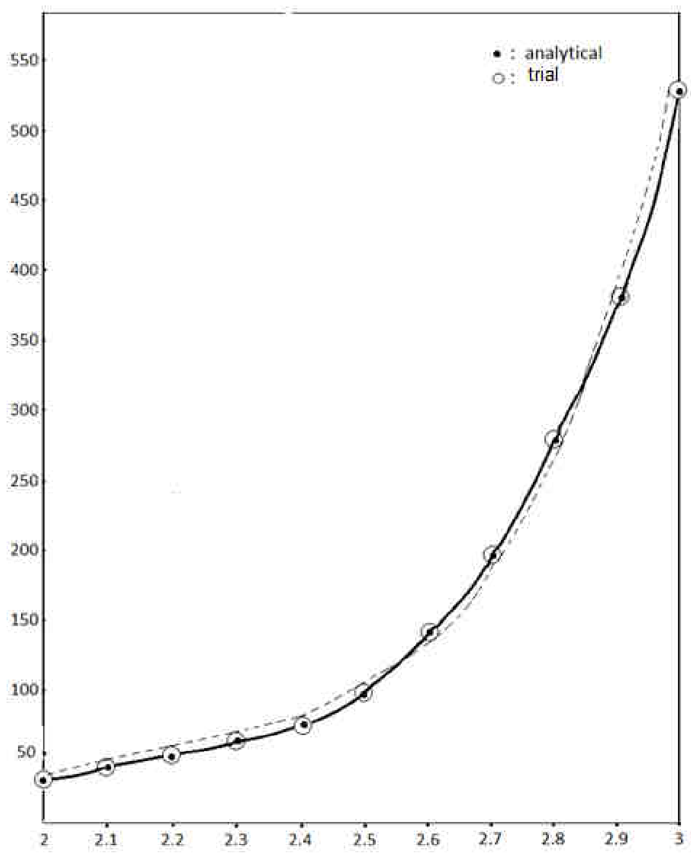

Figure 2. Analytical and trial solutions for example(8.1), for x = 2.5 and y ∈[2,3].

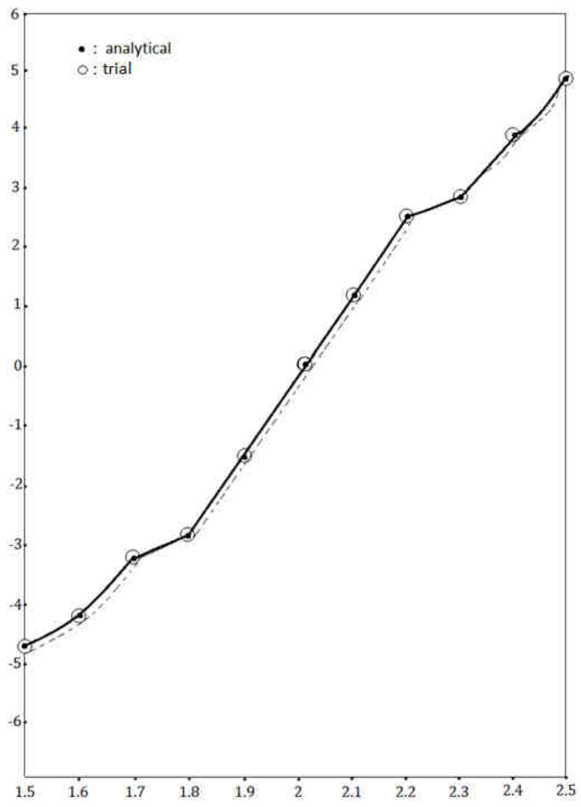

Figure 3. Analytical and trial solutions for example(8.1), for y = 1.5 and x ∈[1.5,2.5].

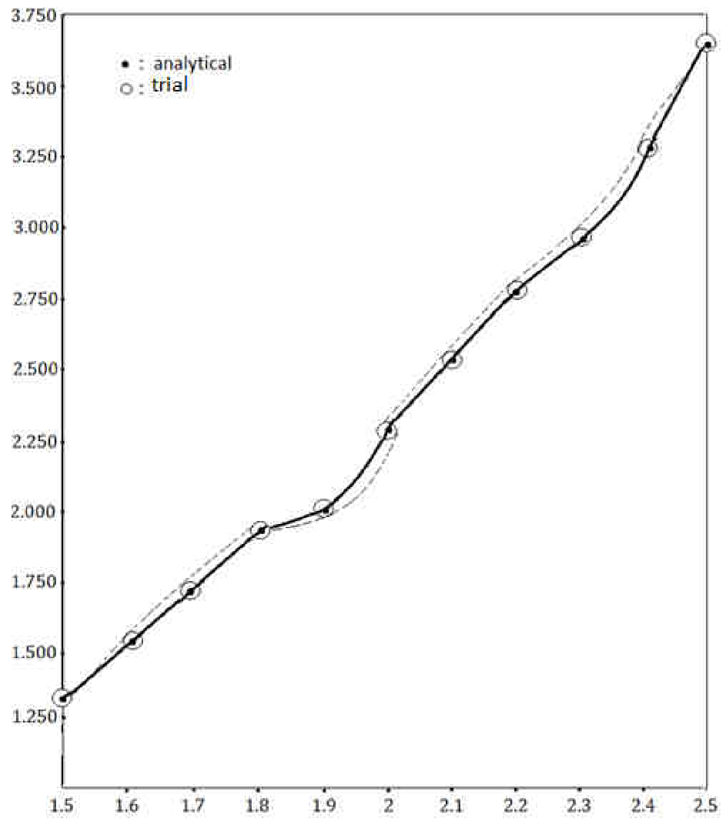

Figure 4. Analytical and trial solutions for example(8.2), for x = 2.2 and y ∈[1.5,2.5].

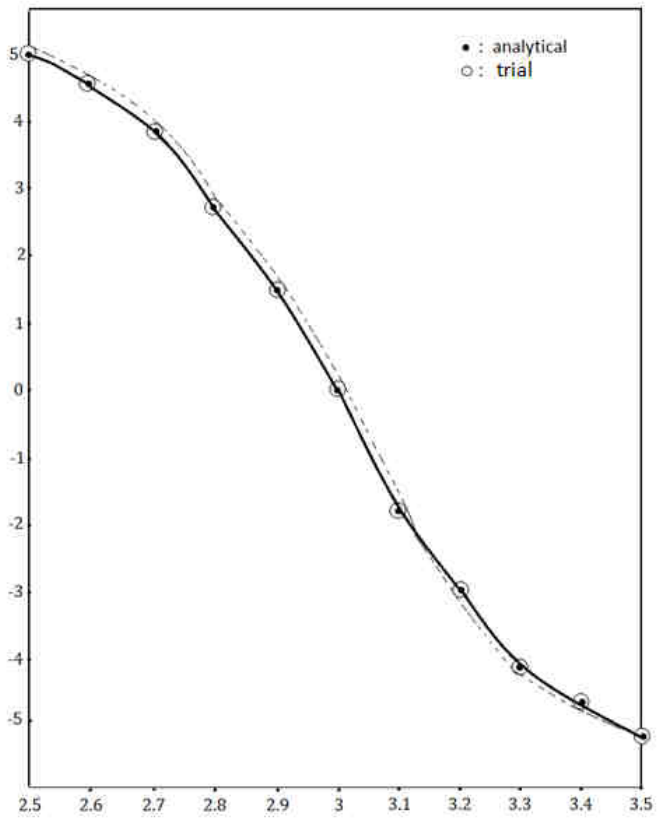

Figure 5. Analytical and trial solutions for example(8.2), for y = 2.2 and x ∈[2.5,3.5].

Table 1. Numerical results for example (8.1).

x	y	u_a (x,y)	by UANN u_t (x,y)	error	by MANN u_t (x,y)	error
0	0	0	0	0	0	0
0.1	0.1	0.008537828	0.008525483	0.000012345	0.008537786	0.000000042
0.2	0.2	0.034097183	0.034074725	0.000022458	0.034097144	0.000000039
0.3	0.3	0.076183058	0.076231065	0.000048007	0.076183619	0.000000561
0.4	0.4	0.132865960	0.132947134	0.000081174	0.132866552	0.000000592
0.5	0.5	0.199152886	0.198921655	0.000231231	0.199153514	0.000000628
0.6	0.6	0.264808483	0.264353083	0.000455400	0.264809191	0.000000708
0.7	0.7	0.311834910	0.311176780	0.00065813	0.311835847	0.000000937
0.8	0.8	0.312025528	0.312814326	0.000788798	0.312019907	0.000005621
0.9	0.9	0.225309818	0.225765479	0.000455661	0.225305948	0.000003870
1	1	0	0	0	0	0

Table 2. Numerical results for example (8.2)

x	y	u_a (x,y)	u_t (x,y)	error
0	0	0	0	0
0.1	0.1	0.003090169	0.003087756	0.000002413
0.2	0.2	0.02351141	0.023509708	0.000001702
0.3	0.3	0.072811529	0.072812021	0.000000492
0.4	0.4	0.152169042	0.152169553	0.000000511
0.5	0.5	0.250000000	0.250000873	0.000000873
0.6	0.6	0.342380345	0.342379432	0.000000913
0.7	0.7	0.396418327	0.396418379	0.000000052
0.8	0.8	0.376182561	0.37618252	0.000000041
0.9	0.9	0.250303765	0.250304127	0.000000361
1	1	0	0	0

Table 3. Maximum and minimum deviation from the analytical solution for the MANN and the UANN .

Example No.	Max. error	Min. error	by	Max. error	Min. error	by
8.1.	0.000005621	0.000000039	MANN	0.000788798	0.000012345	UANN
8.2.	0.000002413	0.000000041	MANN	0.000015000	0.00000100	UANN